I. General Remarks Concerning This Response

Claims 1-23 are currently pending in the present application. No claims have been amended in this response. Reconsideration of the claims is requested.

As was already noted by Applicant in the previous response to the previous Office action and not corrected in the current Office action, Applicant notes that the Office action fails to recite some of the dependent claims in the statement of the grounds of rejection, specifically claims 4-9, 15, 16, and 19-21.

Applicant notes that formal drawings for this application were filed on 02/09/2001; however, the THREE previous Office actions have not acknowledged the receipt of those drawings nor indicated whether the drawings were acceptable, even though Applicant has asked for an acknowledgment in the previous responses. Applicant kindly requests an acknowledgment and an indication of acceptability of the previously submitted formal drawings.

II. Remarks directed to "Response to Arguments" section of current Office action

In the previous response to the previous Office action, Applicant argued extensively against the manner in which the rejection was based on Khoyi et al.. In the current Office action, a rebuttal response is presented to the Applicant's previous arguments, but the Office action's rebuttal response does not address the substance and logic of Applicant's arguments. Instead, the rebuttal response merely repeats references to portions of Khoyi et al. that were already

presented within the rejection; the rebuttal response fails to refute Applicant's arguments.

As argued in the previous response, in the system of Khoyi et al., an object manager manages many objects of a particular object type. As clearly shown in the portions of Khoyi et al. that are recited extensively below, the referenced set of features is not similar to the claim elements. In the present invention, a manager object is associated with an application type, and the manager object manages many instances of the application. Hence, manager objects are distinct entities from instances of applications. In Khoyi et al., an object manager manages various types of objects and does not manage instances of other application programs as required by the claim language.

Because the final rejection remains confusing on the correspondence between features in the system of Khoyi et al. and the present invention, Applicant conjectures that the position of the rejection is that an object manager in the system of Khoyi et al. is similar to a manager object in the present invention.

However, if this is the case, then there are no entities in the system of Khoyi et al. that correspond to instances of applications in the present invention because an object manager in the system of Khoyi et al. corresponds to an application. An object manager in the system of Khoyi et al. cannot correspond to a manager object in the present invention, which manages instances of applications in the present invention, and also correspond to an instance of an application, as stated within Khoyi et al. itself. In other words, an object manager in the system of Khoyi et al. cannot be both an entity that manages instances of applications and the entities that are being managed, i.e. both the managing

entity and the managed entity, yet this is the illogical position that seems to be asserted by the rejection.

The rebuttal response in the current Office action contains the following statements on page 2. First, the Office action states: "applicant argues the prior art does not teach a managed object associated to each application type." It is important to note that it is unclear whether "managed object" is a typographical error or a misinterpretation of the claim. The claim often refers to a "manager object", not a "managed object". While Khoyi et al. clearly discloses "managed objects", Khoyi et al. does not disclose a "manager object" as required by the claim language. If this error was not typographical in nature, then the error may be relevant for understanding why the rejection incorrectly argues that the managed objects in Khoyi et al. are equivalent to the instances of applications in the present invention.

Next, the Office action states: "Examiner notes the prior art taught object manager operates not limited to [sic] the primary object manager for that object type {Khoyi col 10 lines 9-22] or unlimited variety of object types [Khoyi col 9 lines 45-60];". Although the sentence is grammatical incorrect and difficult to understand, Applicant again emphasizes that in the present invention, the manager object manages instances of applications, not merely objects. As explained in the previous response to the previous Office action and again explained in detail hereinbelow, a particular element in Khoyi et al. must be analogous to the manager object of the present invention, and once a choice is made as to which element in Khoyi et al. corresponds to the manager object of the present invention, the argument in the rejection then contradicts itself. Unfortunately, the rejection and the rebuttal response both lack any explanation of a

correspondence between specific elements in the claim language and specific features in Khoyi et al..

The Office action then states: "object management operations can performed ion file based objects [sic] such as: create object, delete object, copy objected. [Khoyi col 15 lines 53-65];". This statement is irrelevant. These "file based objects" are not instances of an application type as required by the claim language, and the entity that performs the management operations in Khoyi et al. is not equivalent to the entity that performs management operations in the present invention.

The rebuttal response in the Office action then continues: "if the copy data of a type that can not be stored in the destination object, then a new object of a type that can stored [sic] the data will be created [Khoyi col 16 lines 34-51]." Again, this statement is irrelevant because the underlying objects in Khoyi et al. are not equivalent to the application instances in the present invention.

The Office action further states: "the association between object type, operation to be performed, and corresponding object manager is performed through the object manager table [Khoyi col 3 lines 15-30, col 35 line 25-col 36 line 5]." Again, the rebuttal response is merely pointing to long sections of Khoyi et al. in a manner similar to the rejection, but the referenced sections do not show the claimed features. It is entirely unclear what point is being made by noting the object manager table.

The rebuttal response in the Office action then concludes:

> As per claims 1, 10, 17, 22, 23, applicant argues the prior art does not teach managing all instances of the application through the manager object. Examiner notes the prior art taught the object manager handles the

request from application by create [sic] a new process running an instance of the need [sic] [Khoyi col 34 lines 35-45]. It is obvious the object manager can create a new process for any or all instances of the application.

Not only is the statement grammatically incorrect, it is entirely confusing. It is unclear what is meant by "the request from application"; more particularly, it is unclear what request is being referenced and, more importantly, which application is being referenced. Applicant argues that this statement is representative of the lack of specificity in the rejection because the rejection lacks a sufficient explanation as to how an analogy can be drawn between the managed objects in Khoyi et al. and the application instances in the present invention. As noted above, these arguments by Applicant are more fully developed hereinbelow.

III. Summary of Present Invention

The present invention is a method for managing a given application type and, in particular, all instances of that application type. A manager object or entity is created to represent each application type running on a client. The manager object may reside anywhere in the distributed network, and the manager object is the target of management operations, which are then redirected to the appropriate client node. The distributed data processing system preferably includes a dataless management framework, also called a lightweight client framework (LCF). One advantage of the present invention is that a server management operation may be directed to a particular type or a given application instance without exposing the remainder of the client nodes to the operation.

The manager object preferably comprises a control routine and a registry containing a set of one or more elements in which each element is a dataset of context-specific information for a given application instance. The context information may include client node identity, installation location, e.g., a directory, installation identifier, e.g., database server name, or other administrative details. When a management operation, such as a query, is performed by a management server, the manager object redirects the query to the client node after possibly augmenting it with appropriate context information. Upon receipt of the query at the client, a query agent is started, and the context information is then used by a local query agent to identify which of the many installed instances of the application to target for the management operation.

IV. 35 U.S.C. § 103(a)—Obviousness—*Khoyi et al. in view of Jeffords et al.*

The Office action has rejected independent claims 1, 4-10, 15-17, and 19-23 under 35 U.S.C. § 103(a) as unpatentable over by Khoyi et al., "Matchmaker for Assisting and Executing the Providing and Conversion of Data Between Objects in a Data Processing System Storing Data in Typed Objects Having Different Data Formats", U.S. Patent No. 5,261,080, filed 08/28/1992, issued 11/09/1993 (hereinafter Khoyi et al.), in view of Jeffords et al., "Replicated Resource Management System for Managing Resources in a Distributed Application and Maintaining a Relativistic View of State", U.S. Patent No. 6,233,623, filed 01/11/1996, issued 05/15/2001 (hereinafter Jeffords et al.). This rejection is respectfully traversed.

Independent claim 1

The Office action has rejected independent claims 1, 10, 17, 22, and 23 using a common argument that focuses on the claim elements of independent claim 1. In other words, the arguments in the rejection focus on the elements of claim 1, which reads:

> 1. A method of managing a set of clients in a distributed computer network having a management server, comprising the steps of:
>
> associating a manager object to each application type on a given client, the manager object including a registry having a set of one or more elements, wherein each element includes information representing a context of an application instance; and
>
> managing all instances of the application through the manager object.

Rejection of claims 1, 10, 17, 22, and 23

The rejection of claims 1, 10, 17, 22, and 23 reads as follows in its entirety:

> As per claims 1, 10, 17, 22, 23 Khoyi discloses a method of managing a set of clients in a distributed computer network having a management server, comprising the steps of:
>
> associating a manager object to each application type on a given client (col 9 lines 38-col 10 line 29, col 13 lines 30-52]; and
>
> managing all instances of the application through the manager object which is equivalent to the object managers are peer with each other and are children of the application manager [col 14 lines 40-45]. Khoyi also taught the object catalog for a link registry [col 70 lines 15-32]. However Khoyi is silent on the manager object including a registry having a set of one or more elements wherein each element includes information representing a context of an application instance. A skilled artisan would have motivation to improve the manager object on Khoyi's apparatus and found Jeffords teaching. Jeffords discloses a manager object including a registry having a set of one or more elements such as manager object having an associated set of memory pools and a registry of the network unique identifiers for the resource objects in the associated set of memory pools

where each pool representing an RRM process which is equivalent to the element includes information representing a context of an application instance [col 10 lines 30-43, col 11 lines 20-62]. Khoyi-Jeffords also taught redirecting the modified query to the client machine [Khoyi col 43 lines 33-55] [Jeffords col 15 lines 47-67].

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate the registry with a set of memory pools as taught by Jeffords into the Khoyi's apparatus in order to improve the manager object. Doing so would provide a simple, dynamic and efficient process to the manager object of the distributed computing system.

Thus, the system and method of claims 1, 10, 17, 22 and 23 is obvious in view of the combination of references.

Analysis of the rejection

The rejection of the claims under Khoyi et al. and Jeffords et al. starts with an attempt to build a foundation for an obviousness argument with multiple citations to passages of Khoyi et al.; these passages are applied against a portion of the first element of claim 1. However, none of these passages disclose the claimed features of the independent claims. When the rejection turns to Jeffords et al. as a secondary reference for disclosure of other claim elements that are admittedly not shown in Khoyi et al., Jeffords et al. suffers from the same deficiencies, i.e. none of these passages disclose the claimed features of the independent claims.

More specifically, the rejection merely applies portions of Khoyi et al. and Jeffords et al. against claim elements without careful consideration of the claim language and the subject matter. Passages from the references appear to have been cited in the rejection only because the passages included terminology that seemed similar to the claim language. The form of the rejection appears appropriate, but the argument

presented by the rejection is illogical because the basis of the rejection, i.e. the cited portions of the reference, do not disclose what the rejection purports that they disclose. More importantly, the references fail to disclose the claimed elements of the present invention.

The rejection addresses the first portion of the first element of independent claim 1 with reference to two portions of Khoyi et al., which together provide a sufficient description of the operation of the disclosed system and state the following (emphasis added):

> The system of the present invention is a member of the general class of systems described as "object based". That is, information is stored in structures referred to as "objects". In the implementation of the present preferred embodiment most objects each correspond to one or more files of a conventional computer file system.
>
> Further with regard to objects, the system of the present invention allows the use of an essentially unlimited variety of object "types", including the type previously described as a folder, wherein there may be a type of object for each form of data or information or operation to be performed by the system. That is, the system of the present invention defines only the minimal interface between an object and the system and does not define the internal structure or form of any object. As such, an object within the system of the present invention may be regarded as a general purpose container for data, programs or other information, with the internal structure or form of a particular object being defined by the requirements of the operation to be performed or the type of data or information to be stored therein.
>
> **Programs for operating upon objects are known as "object managers" or are sometimes referred to as "editors", "applications programs", or "applications". The term "application" is also used to refer to a collection of object managers that operate on a single object type. Each type of object has associated with it at least one object manager that is designed or intended as the primary means for operating upon the data or information stored in that type of object.** For example, the system may support a "document type" object for word

processing and there will be a word processing object manager associated with that object type. Similarly, a "data base type" object will have associated with it a data base object manager, which is the primary means for operating upon or with the data stored in the data base type object.

It should be noted, however, that the object managers for operating with a particular type of object are not limited to the primary object manager for that object type. Moreover, the particular object manager invoked to operate upon a particular object may depend upon the type of operation to be performed. The system of the present invention provides for a plurality of object managers to operate with any given object type and certain object managers, for example, certain utilities, may operate with more than one type of object. The primary object manager is simply the object manager which is invoked by default for operations upon a particular object type if the user does not select a different program.

Although typically an object manager will operate on the data of a single type of object, in certain cases it may be desirable to arrange a program to be an object manager for more than one object type. Further, there are various utility programs that perform operations that do not interpret object data (e.g. file copy) and that therefore can be used with various types of objects. --(Khoyi et al., col. 9, line 38, to col. 10, line 29).

For each type of object there is one or more "object manager" that can operate on objects of that type. **Object managers roughly correspond to applications programs.** For example, there may be spreadsheet program: this can be understood to be an "application", as distinguished from operating system program; it can also be understood to be a "manager" or "editor" of objects of type "spreadsheet".

Because it is an object's object manager(s) that define its structure and interpretation, the collection of object types is open-ended; existing types of information can be integrated with future types of information with the same each with which the existing types are integrated with each other. A new object type can be added to the system by adding a program (i.e., an object manager) that can manipulate objects of the new type. The ability to manipulate the new type of object need only be implemented once. Having added the new

> object manager, objects of the new type can be linked into, exchange data with, and otherwise appear integrated with objects of pre existing types without modification to pre existing object managers.
> --(Khoyi et al., column 13, lines 30-52).

In the system of Khoyi et al., an object manager manages objects of a particular object type. As clearly shown in the recited portions of Khoyi et al., **object managers correspond to application programs**.

This set of features is not similar to the claim elements against which they are cited. In the present invention, a manager object is associated with an application type, and the manager object manages instances of the application. Hence, manager objects are distinct entities from instances of applications.

The rejection is confusingly silent on the correspondence between features in the system of Khoyi et al. and the present invention. The rejection merely cites a section of Khoyi et al. without taking a position on the which entities in the system of Khoyi et al. might be equivalent to entities in the present invention. Applicant conjectures that the position of the rejection is that an object manager in the system of Khoyi et al. is similar to a manager object in the present invention.

However, if this is the case, then there are no entities in the system of Khoyi et al. that correspond to instances of applications in the present invention because an object manager in the system of Khoyi et al. corresponds to an application. An object manager in the system of Khoyi et al. cannot correspond to a manager object in the present invention, which manages instances of applications in the present invention, and also correspond to an instance of an application, as stated within Khoyi et al. itself. In other

words, an object manager in the system of Khoyi et al. cannot be both an entity that manages instances of applications and the entities that are being managed, i.e. both the managing entity and the managed entity, yet this is the illogical position that seems to be asserted by the rejection.

The rejection then addresses the second element of independent claim 1 with reference to a portion of Khoyi et al., which states the following (emphasis added):

> In the present application, there is an Application Manager process. The Application Manager spawns the object manager processes. Thus, in general, the object managers are peers with each other and are children of the Application Manager.
> --(Khoyi et al., column 14, lines 40-45).

The only way that this section of Khoyi et al. could be logically interpreted as being relevant to the claim language is that the Application Manager in the system of Khoyi et al. is somehow analogous to a manager object in the present invention. However, the rejection has already taken a position as to which entity in the system of Khoyi et al. is supposedly equivalent to a manager object in the present invention. Hence, the rejection appears to assert contradicting positions.

Moreover, the Application Manager does not have the characteristic of being associated with any one particular application type; the Application Manager manages all object managers in the system of Khoyi et al.. Hence, an assertion that the Application Manager is supposedly equivalent to a manager object in the present invention also fails with respect to the claim language because the claim states: "associating a manager object to each application type".

Contrary to the statements in the rejection, Khoyi et al. fails to show the first portion of the first element of claim

1 and the second element of claim 1. The rejection then turns to Jeffords et al. and states that Jeffords et al. discloses the second portion of the first element of claim 1, i.e. "the manager object including a registry having a set of one or more elements, wherein each element includes information representing a context of an application instance". The rejection asserts that Jeffords et al. teaches this feature in two sections of the reference, which recited hereinbelow. However, these sections in Jeffords et al. are recited not for ease of reference for the reader but to support Applicant's contention that these sections are irrelevant to the claimed feature.

> Replication of an object's construction occurs when an object is first created within a resource pool, and when a resource pool is synchronized. Replication of an object's attributes occurs incrementally as the attributes' state changes. Both of these replication functions require an object transport function.
>
> The RRM provides a means by which software objects are transported between processes. The object transport function is able to take any "supported" software object and convert it to a transportable (distributable) representation that may be sent to and received from any RRM processes. Supported software objects are objects that can be distributed (streamed).
>
> Thus, the object transport function obtains an object's distributable representation from that object and creates an object from its distributable representation; this is accomplished by an object packetization service, also known as streaming and unstreaming, or packing and unpacking.
> --(Jeffords et al., column 10, lines 30-43).

> When an RRM process enters a distributed computing function it must send a join pool request to all active RRM processes for each resource pool it is interested in. Each one of the active processes must respond with either a positive acknowledgment (meaning it is also interested in the pool specified), or a negative acknowledgment (it is not interested). Each of the positively responding processes must then send all of the objects it is responsible for ("owns") in the pool to the process

joining the pool (when a pure in memory replication scheme is used). If the process joining the pool already has objects in that pool that it is responsible for, it must send those objects to each of the processes that responded positively. In this way, each pool is synchronized in both directions for every process that is interested in the pool.

A separate level of synchronization and consistency checks occurs at the memory level. This ensures that: (1) a consistent view of memory pool existence and RRM process attachment is maintained (i.e., what processes have joined what pools); (2) each pool's objects are distributed and maintained consistently; and (3) the relative state of pool synchronization is distributed and checked.

Memory level consistency checks include:
agreement as to which RRM processes are interested in each pool; and
relativistic synchronization of each pool.

Memory level synchronization includes:
distribution of all objects created within a pool to all RRM processes interested in (joined to) the pool (synchronization of object existence); and
bidirectional pool synchronization and agreement upon the pool's synchronization state between all RRM processes joined to each pool.

Memory level synchronization is achieved when:
every resource pool an RRM process is interested in is synchronized with respect to the given RRM process; a resource pool is synchronized with respect to a given RRM process when all other RRM processes have sent their owned objects to the given RRM process and the given RRM process has sent its owned objects to all other interested RRM processes.

--(Jeffords et al., column 11, lines 20-62).

It is entirely unclear why these particular sections of Jeffords et al. are referenced by the rejection.

The rejection states that Jeffords et al. discloses "a manager object having an associated set of memory pools and a registry of the network unique identifiers for the resource

objects in the associated set of memory pools where each pool representing an RRM process". Elsewhere, Jeffords et al. states the following, which seems to contain the feature that is stated within the rejection:

> [E]ach RRM [Replicated Resource Management] process has an associated resource manager (RM) which identifies several resource pools of interest, i.e., several meetings which are of interest to the person. In this way, a resource manager (process) participates in just its areas (pools) of interest, maintains a registry of resources in these associated pools, and includes mechanisms for communicating with other resource managers (processes) in order to maintain a consistent view of the state of the resources in the associated pools. --(Jeffords et al., column 2, lines 32-41).

However, Applicant asserts that the feature of resource pools or memory pools in the system of Jeffords et al. is not equivalent to the claim element against which this feature is compared, namely, "the manager object including a registry having a set of one or more elements, wherein each element includes information representing a context of an application instance". More importantly, the feature of a context of an application instance is not present at all in Jeffords et al..

As shown by Applicant in the arguments above, Khoyi et al. and Jeffords et al. do not disclose the claim elements as stated in the rejection. The rejection then proceeds to state that it would have been obvious to combine features from the system disclosed in Khoyi et al. and the system disclosed in Jeffords et al.. However, the motivational statement in the rejection is completely generic: "... in order to improve the manager object. Doing so would provide a simple, dynamic and efficient process to the manager object of the distributed computing system." The rejection does not provide any argument as to how or why a characteristic of the system of

Khoyi et al. or a characteristic of the system of Jeffords et al. would motivate one of ordinary skill in the art to combine any feature of these systems with any feature of the other system. Applicant admits that, in general, one of ordinary skill in the art would desire an "efficient process" and would want to "improve" a system. However, Applicant asserts that the rejection completely fails to provide a reason why one would be motivated to combine features from the prior art references.

The rejection does not explicitly address independent claims 17 and 22, but the rejection does state: "Khoyi-Jeffords also taught redirecting the modified query to the client machine [Khoyi col 43 lines 33-55] [Jeffords col 15 lines 47-67]." Apparently, this statement addresses the following feature in claim 17 (claim 22 has a similar element):

> a control routine (a) for intercepting a query directed to the client machine, (b) for modifying the query as a function of the information; and (c) for redirecting the modified query to the client machine to target management of the application instance.

The section of Khoyi et al. that is cited by the rejection is recited below; again, this section is recited not for ease of reference for the reader but to support Applicant's contention that this section is irrelevant to the claimed feature.

> When the source data is similar to or identical to the destination data, either directly or after conversion, the copied, moved or shared source data will "blend into" or become "embedded" in the destination data. In the case of copied or moved data, the copied or moved data may be editable by the destination object manager. In the case of shared data, however, and while the shared data may be copied, moved or deleted, it may be edited only in the source object and only by an object manager for the source object. The destination object's

> object manager may convert the shared data into its own internal data format but may not incorporate the shared data into its own data, may not edit the source data in any way, and is required to "mark and hold" the shared data. The data should be visually marked for the user, so the user is aware that the data is shared, rather than directly a part of the object being edited. In the case of data that is "marked and held", an entry in the link parallel file of the destination object will contain (i.e., "hold") a copy of the linked data. This copy in the link parallel file cannot directly be edited: rather, this data is edited by editing the source object from which it is linked.
> --(Khoyi et al., col. 43, lines 33-55).

It is entirely unclear why this particular section of Khoyi et al. has been cited by the rejection as disclosing a feature concerning "queries"; it is clear that this particular section is completely irrelevant to the claimed feature.

The section of Jeffords et al. that is cited by the rejection against the feature in independent claims 17 and 22 is recited below; again, this section is recited not for ease of reference for the reader but to support Applicant's contention that this section is irrelevant to the claimed feature.

> The embodiment described herein assumes a communications network is modelled in memory, such as cache memory in a network manager. The network manager may be a Sun workstation sold by Sun Gateway 2000 with a Sun Solaris Microsoft Windows NT operating system; the network management software may be the Spectrum.TM. Network Management software sold by Cabletron Systems, Inc., Rochester, N.H., USA and described in U.S. Pat. No. 5,261,044 by R. Dev et al., which is hereby incorporated by reference in its entirety. Typically, the network manager resides in a host and the resources of a network provide alerts to the host which are received and interpreted by the network manager in accordance with known techniques. The resources of a network may include a plurality of individual workstations, personal computers, modems, applications resident in host computers, communication satellites, and the like. In a

> modelled network, a change made by an application to one resource of the network, which affects the other resources of the networks, will be immediately known and accounted for since the model of the network within the data cache manager accounts for these inter-relationships.

Again, it is entirely unclear why this particular section of Jeffords et al. has been cited by the rejection as disclosing a feature concerning "queries"; it is clear that this particular section is completely irrelevant to the claimed feature.

As shown by Applicant in the arguments above, Khoyi et al. and Jeffords et al. do not disclose the claim elements of the independent claims. Applicant maintains that the arguments that have been given above for the patentability of the independent claims also apply to their dependent claims. Applicant also provides some additional arguments for the patentability of some of the dependent claims.

With respect to dependent claim 4 (and similar claim 19), the rejection states that the claim, which reads "the element includes information identifying a client node", is shown in the system of Khoyi et al. as a user logon ID. Applicant asserts that an identifier for a user is not equivalent to an identifier for a client device.

With respect to dependent claim 5 (and similar claim 20), the rejection states that the claim, which reads "the element includes information identifying the directory where the application instance is installed", is shown in Jeffords et al. as a directory. However, Jeffords et al. merely states at column 14, lines 59-61: "A distributed memory space can be divided into separate pools for each of the following services: topology, directory, policy, and call". At most, Jeffords et al. merely mentions a directory; Jeffords et al.

does not disclose "a registry having an element with information representing a context of an application instance wherein the information includes a directory where the application instance is installed", as is required by the claim language of independent claim 1 together with the claim language of dependent claim 5.

With respect to dependent claim 7, the rejection states that the claim, which reads "the application type is discovered by the manager object", is shown in Jeffords et al. by an RRM discovery process. However, Jeffords et al. states: "During this discovery process, each RRM process learns about the relative state of all other RRM processes in the network."--(Jeffords et al., col. 5, lines 22-24). As already discussed above, Khoyi et al. was used by the rejection as a reference that disclosed something about application types, so it is unclear how Jeffords et al. now discloses something about discovering an application type. More importantly, this passage from Jeffords et al. clearly does not disclose the claim element, which requires "the application type is discovered by the manager object".

With respect to dependent claim 8, the rejection states that the claim, which reads "further including the step of discovering the application type prior to associating the manager object", is shown in Jeffords et al. by the same RRM discovery process that was mentioned above with respect to dependent claim 7. As argued above, Jeffords et al. does not disclose the claimed features.

Examiner bears the burden of establishing a *prima facie* case of obviousness.

The examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when

rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Only when a *prima facie* case of obviousness is established does the burden shift to the applicant to produce evidence of nonobviousness. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Rijckaert*, 9 F.3d 1531, 1532, 28 U.S.P.Q.2d 1955, 1956 (Fed. Cir. 1993). If the Patent Office does not produce a *prima facie* case of unpatentability, then without more the applicant is entitled to grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grabiak*, 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985). In response to an assertion of obviousness by the Patent Office, the applicant may attack the Patent Office's *prima facie* determination as improperly made out, present objective evidence tending to support a conclusion of nonobviousness, or both. *In re Fritch*, 972 F.2d 1260, 1265, 23 U.S.P.Q.2d 1780, 1783 (Fed. Cir. 1992).

With respect to claims 1, 4-10, 15-17, and 19-23, Applicant respectfully submits that claimed features are not shown in the prior art references nor can the teachings of the references be combined to disclose the present invention. Hence, the rejection of claims 1, 4-10, 15-17, and 19-23 does not establish a *prima facie* case of obviousness based on the prior art. Therefore, the rejection of claims 1, 4-10, 15-17, and 19-23 under 35 U.S.C. § 103(a) has been shown to be insupportable, and these claims are patentable over the applied references. Applicant requests that the rejection be withdrawn.

V. 35 U.S.C. § 103(a)–Obviousness–*Khoyi et al.* in view of *Jeffords et al.* and further in view of *Bereiter*

The Office action has rejected claims 2, 3, 11, 12, and 18 under 35 U.S.C. § 103(a) as unpatentable over Khoyi et al. in view of Jeffords et al. and further in view of Bereiter, "Software Auditing Mechanism for a Distributed Computer Enterprise Environment", filed 10/01/1996, issued 05/19/1998. This rejection is respectfully traversed.

With respect to dependent claim 2 (and similar claim 11), the rejection properly states that Bereiter discloses a dataless management framework. However, claim 2 depends from claim 1, and as argued above, Khoyi et al. and Jeffords et al., either singly or in combination, fail to disclose the features of claim 1. Hence, a combination of the teaching of Bereiter with Khoyi et al. and Jeffords et al. cannot support a rejection of dependent claim 2.

With respect to dependent claim 3 (and similar claims 12 and 18), the rejection states that the feature of "wherein the dataless management framework includes a local agent that is controlled by the manager object to manage the application instance" is an inherent feature of a dataless management framework. First, Applicant notes that Khoyi et al. nor Jeffords et al. nor Bereiter disclose the feature of using a manager object to manage an application instance as claimed. Second, Applicant asserts that the teachings of three references that do not separately disclose a claimed feature cannot be combined to "inherently" produce a claimed feature. The rejection must explain which features of the teachings of each reference are being combined as support for an obviousness-type argument.

Examiner bears the burden of establishing a *prima facie* case of obviousness.

With respect to claims 2, 3, 11, 12, and 18, Applicant respectfully submits that claimed features are not shown in the prior art references nor can the teachings of the references be combined to disclose the present invention. Hence, the rejection of claims 2, 3, 11, 12, and 18 does not establish a *prima facie* case of obviousness based on the prior art. Therefore, the rejection of claims 2, 3, 11, 12, and 18 under 35 U.S.C. § 103(a) has been shown to be insupportable, and these claims are patentable over the applied references. Applicant requests that the rejection be withdrawn.

VI. Conclusion

It is respectfully urged that the present patent application is patentable, and Applicant kindly requests a Notice of Allowance.

For any other outstanding matters or issues, the examiner is urged to call or fax the below-listed telephone numbers to expedite the prosecution and examination of this application.

DATE: March 5, 2003

Respectfully submitted,

Joseph R. Burwell
Reg. No. 44,468
ATTORNEY FOR APPLICANT

Law Office of Joseph R. Burwell
P.O. Box 28022
Austin, Texas 78755
Voice: 866-728-3688 (866-PATENT8)
Fax: 866-728-3680 (866-PATENT0)
Email: joe@burwell.biz